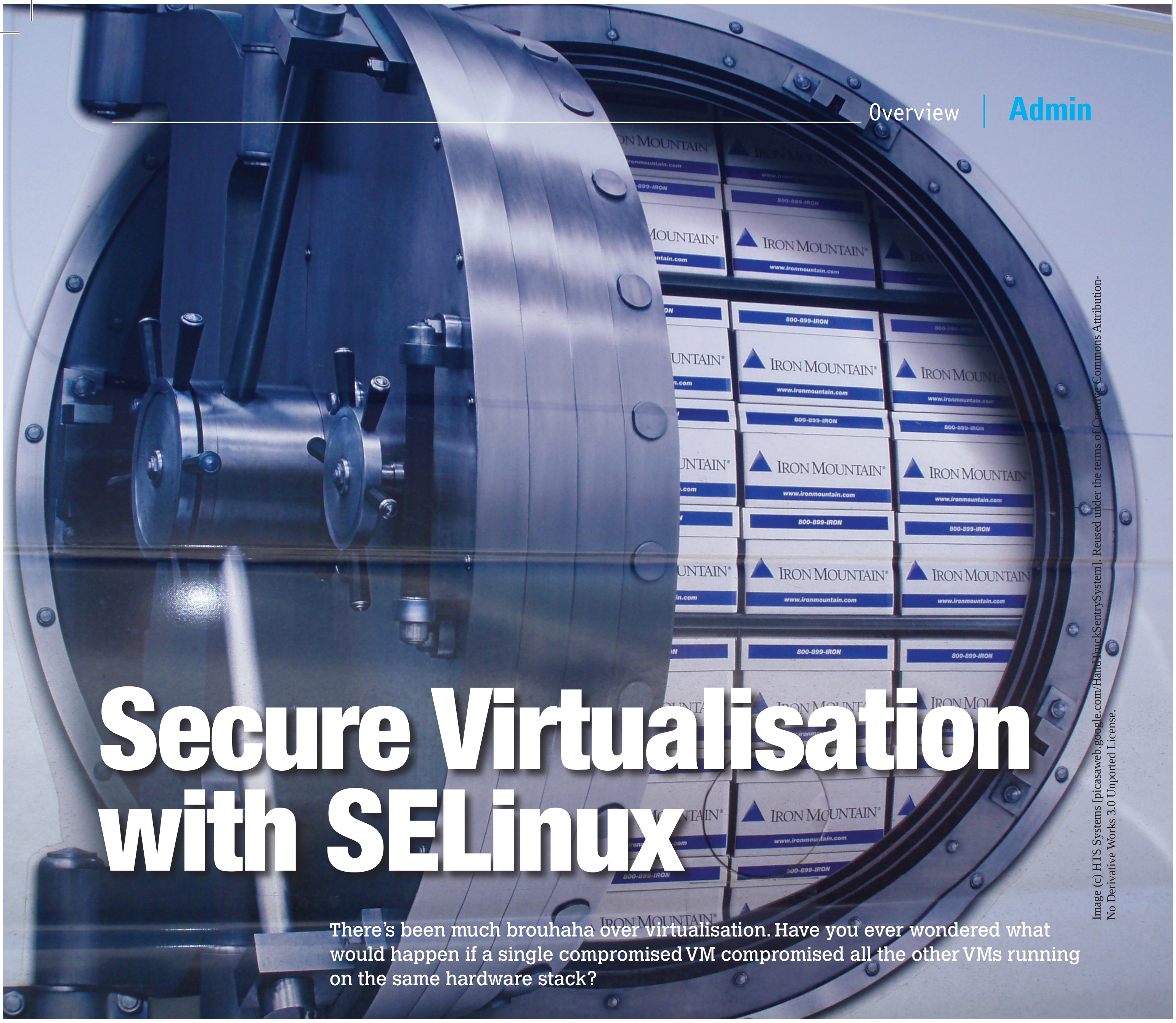

Secure Virtualisation with SELinux

There's been much brouhaha over virtualisation. Have you ever wondered what would happen if a single compromised VM compromised all the other VMs running on the same hardware stack?

As the usage of virtualisation increases, the security that isolated physical machines enjoy is at risk. Rogue processes in a guest OS might break out of the virtual machine and get access to the host memory or other guests. Designing security features within hypervisors can control this. But in case of such incidents, the risks can be mitigated by integrating Mandatory Access Controls (MAC) with virtualisation.

With the integration of MAC-capable security schemes like SELinux, with virtualisation, the security position of guest machines can be improved.

Let's take a look at how the scenario might unfold. We can begin by examining the situation as it existed in the good old days when one physical machine ran a dedicated server:

- Before virtualisation, servers were generally physically isolated from each other (Figure 1). An attacker taking over one server would typically have control over just that. So, if he wanted to breach the other machines on the network, he essentially had to launch a network-based attack to exploit them. This limited the damage of network attacks. To avoid these kinds of attacks, administrators used different tools like network firewalls, intrusion detection/prevention systems, and unified threat management systems to mitigate the risks.
- With the advent of virtualisation, several guest machines run on the same single host server (Figure 2), and probably as the same UID. This comes with its own risks. If an attacker gains access to a single virtual

machine, he then just needs to breach the hypervisor running on the host. If there is a hypervisor vulnerability currently existing in the wild, he can exploit it and can thus, eventually, take over the rest of the virtual machines running on the host.

Understanding sVirt

sVirt was introduced as a feature of Fedora 11, and conceived as a security mechanism for Linux-based virtualisation schemes like the (KVM)/Qemu, and lguest. It has been developed primarily by Dan Walsh and James Morris, of Red Hat. Some of the goals/use-cases of sVirt are listed below:

- Providing virtualised services with a level of isolation similar to that previously afforded by physical separation.
- Increased protection of the host from untrustworthy VM guests (for example, for VM hosting providers, grid/cloud servers, etc).
- Increased protection of VM guests from each other in the event of host flaws or misconfiguration. Some protection may also be provided against a compromised host.
- Strongly isolating desktop applications by running them in separately labelled VMs (for example, online banking in one VM and *World of Warcraft* in another; opening untrustworthy office documents in an isolated VM for view/print only).

How does it work?

sVirt essentially works by adding SELinux support to the Linux virtualisation library libvirt. It is designed as a pluggable security framework into the libvirt API (Figure 3). It also supports other MAC security schemes like SMACK (Simplified Mandatory Access Control Kernel), a security kernel module. In a typical use, one ought not to even notice sVirt running in the background.

As you already know, SELinux works by labelling everything like files, devices, processes, etc. The libvirt daemon (*libvirtd*) starts all virtual machines. And all virtual machines run as separate processes. libvirtd dynamically generates labels for the image files and starts the virtual machines with the appropriate labels. The SELinux policy has rules that allow the *svirt_t* processes to read/write *svirt_image_t* files and devices.

This allows us to protect the host machine from any of its VM guests. A virtual machine will only be able to interact with the files and devices with the correct labels. A compromised virtual machine would not be allowed to read my home directory, for example, even if the virtual machine is running as the root.

However, this ‘type’ protection does not prevent one virtual machine from attacking another. We need a way to label the domains and the image files with the same TYPES. Yet, at the same time, we need to stop virtual machine 1 (running as svirt_t) from attacking virtual machine 2, which would also be running as svirt_t.

This is where the Multi Category Security (MCS) mechanism comes into play. MCS is an enhancement

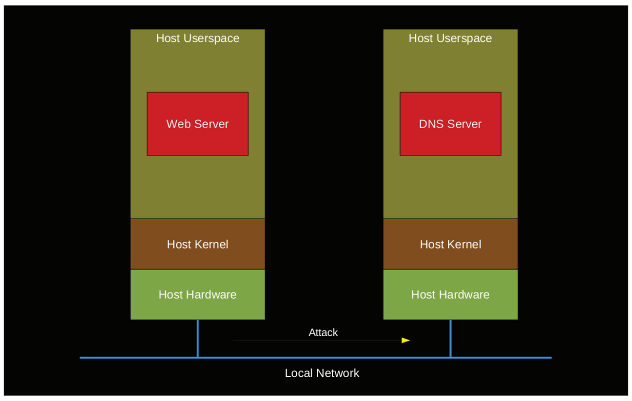

Figure 1: Server set-ups in the absence of virtualisation

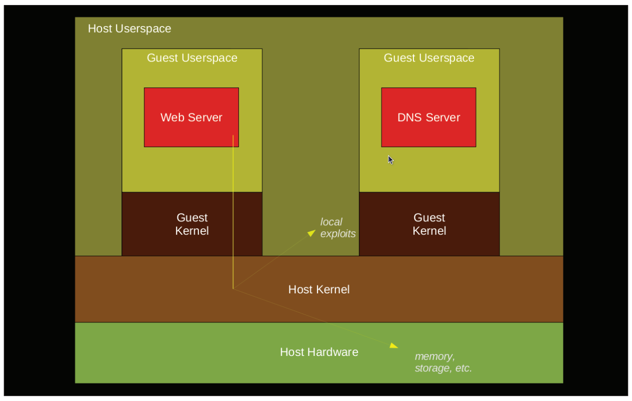

Figure 2: Server set-ups using virtualisation

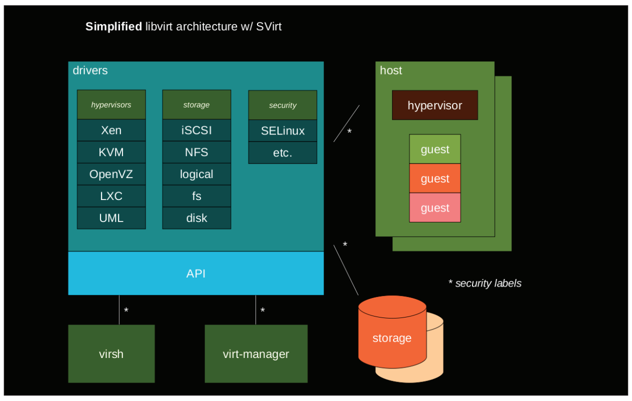

Figure 3: Simplified libvirt architecture with sVirt integration

to SELinux, which allows users to label files with categories. These categories are simply text labels, such as ‘Company_Confidential’ or ‘Medical_Records’. The sysadmin first configures the categories, then assigns users to them, as required. In a nutshell, to access a file, a user needs to be assigned to all of the categories with which the file is labelled.

Note: The first thing the sysadmin will want to do with MCS is configure some categories. This can be done by editing the */etc/selinux/targeted/setrans.conf* file. This file specifies how the internal MCS categories can be mapped to human-

Table 1: Example VMs and their SELinux labels

Name	Virtual Machine process label	Virtual Machine image label
Virtual Machine 1	system_u:system_r:svirt_t:s0:c514,c932	system_u:object_r:svirt_image_t:s0:c514,c932
Virtual Machine 2	system_u:system_r:svirt_t:s0:c101,c230	system_u:object_r:svirt_image_t:s0:c101,c230

readable formats. The following snippet shows the typical contents of the *setrans.conf* file:

```
s0:c0=CompanyConfidential
s0:c1=PatientRecord
s0:c2=Unclassified
s0:c3=TopSecret
```

From the above snippet, the left hand side of the ‘equals to’ sign is the MCS security level, with the corresponding human readable value on the right. To list the current categories defined on your system, use the *chcat -L* command.

Each VM has unique MCS categories, and the SELinux security policy does not allow interaction between different categories, irrespective of their actual values. The file systems' device/image for each VM is also labelled with the VM's MCS label, to prevent VMs from accessing each other’s resources.

Till Fedora Core 5, the typical SELinux context consisted of only three fields: USER:ROLE:TYPE.

Later on, SELinux added one more field, so currently it's: USER:ROLE:TYPE:MCS Label.

Let's look at the example of Table 1, where libvirt creates two virtual machines with dynamically generated SELinux labels.

You can obtain the SELinux labels for a running VM process, as follows:

```
# ps -eZ | grep qemu
system_u:system_r:svirt_t:s0:c514,c932
```

You can obtain the SELinux labels for the image, as follows:

```
# ls -lZ /var/lib/libvirt/qemu/images/f12.img
system_u:object_r:svirt_image_t:s0:c514,c932
```

In the above snippets, take a look at the MCS labels. Here, s0 indicates the 'Sensitivity Level', and c514 as well as c932 indicate the 'category of the data'. Administrators can define these categories.

If you attempt to change the SELinux context (using the chcon command) of the VM image file, it will throw up SELinux AVC (Access Vector Cache) denials.

SELinux prevents virtual machine 1 (system_u:system_r:svirt_t:s0:c514,c932) from accessing virtual machine 2's image file (system_u:object_r:svirt_image_t:s0:c101,c230)—the VMs can't access each other's data, either on-disk or in-memory and hence cannot attack each other. END

References and acknowledgements

- Daniel Walsh: *danwalsh.livejournal.com (sVirt blog posts/presentations/inputs)*
- James Morris: *http://blog.namei.org (sVirt blog posts/presentations/inputs)*
- Amit Shah and rest of Red Hat virtualization team: *Review*
- svirt Wiki: *http://selinuxproject.org/page/SVirt*
- Libvirt Project page: *http://wiki.libvirt.org*

By: Kashyap Chamarthy

Kashyap Charmarthy. The author has been working on security, identity management and PKI technologies for around three years and has a keen interest in Linux-based virtualisation. Currently he is working on open source identity technologies at Red Hat.